EXHIBIT 9

EXHIBIT B

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

Chart Detailing Defendant's Infringement of U.S. Patent No. 9,298,864

Wapp Tech Ltd. & Wapp Tech Corp. v. J.P. Morgan Chase Bank, N.A.,
Case No. 4:23-cv-1137-ALM (E.D. Tex.)

The Accused Instrumentalities include tools from Google used to develop applications for Android mobile devices, including Android Studio, Android Emulator, Android Virtual Devices, Android Profiler, and Android App Inspection tools.

Based on the information presently available to them, Plaintiffs Wapp Tech Limited Partnership and Wapp Tech Corp. ("Wapp" or "Plaintiffs") are informed and believe that Defendant directly and indirectly infringes U.S. Patent No. 9,298,864 (the "'864 Patent"). Defendant directly infringes the '864 Patent when its employees, agents, and/or representatives use the Accused Instrumentalities to test applications for mobile devices. Upon information and belief, to the extent Defendant uses third parties in the testing process, Defendant indirectly infringes the'864 Patent by actively inducing the direct infringement of third parties contracted to use the Accused Instrumentalities to test applications for mobile devices on Defendant's behalf.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

Table of Contents

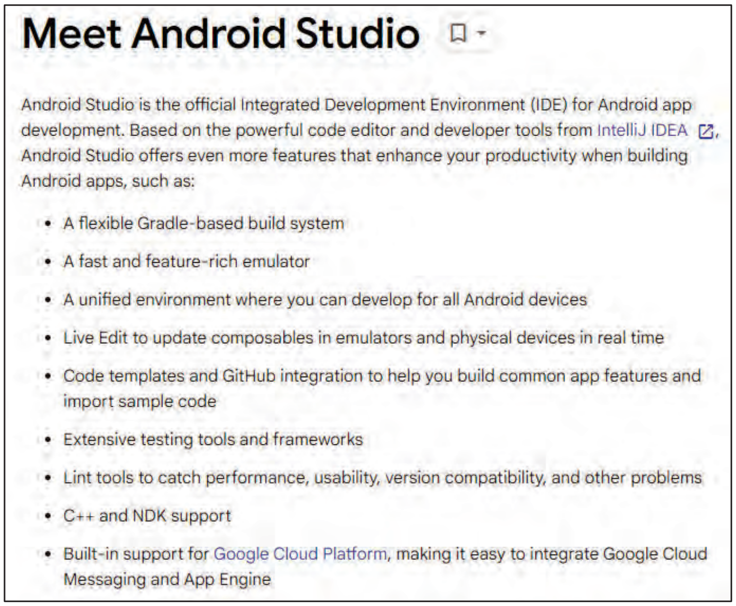
Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[A] A system for testing an application for a mobile device comprising:

Claim 1

1[A] A system for testing an application for a mobile device comprising:

The Accused Instrumentalities are a system for testing an application for a mobile device. Defendant tests its mobile banking applications through its use of the Accused Instrumentalities by executing compiled source code for the application and monitoring the execution of the source code. Android Studio is software for testing applications based on the Android operating system. The Android operating system runs on various mobile devices, including smartphones, tablets, and wearables. Android Studio includes "[a] fast and feature-rich emulator" and "[e]xtensive testing tools and frameworks."



Meet Android Studio

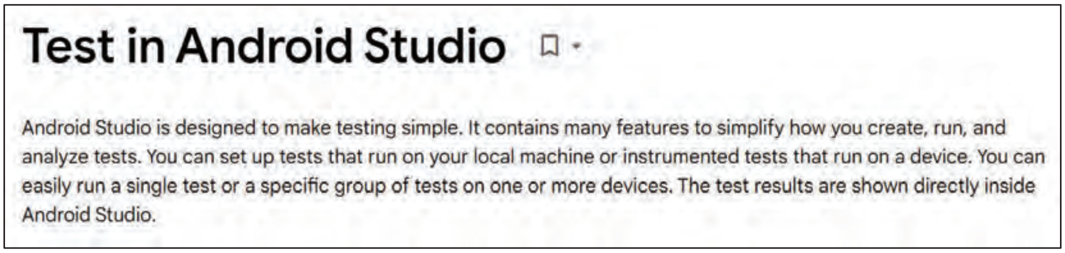
Android Studio is the official Integrated Development Environment (IDE) for Android app development. Based on the powerful code editor and developer tools from IntelliJ IDEA, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Live Edit to update composables in emulators and physical devices in real time
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

https://developer.android.com/studio/intro (last visited 4/6/2024).

<u>Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)</u>

1[A] A system for testing an application for a mobile device comprising:

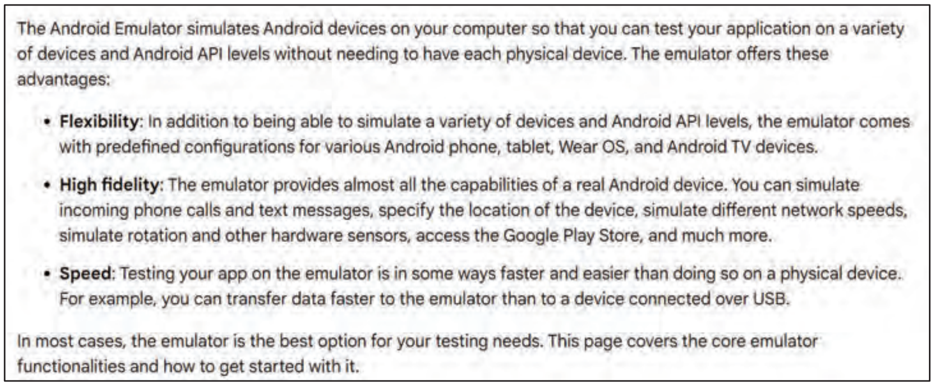


Test in Android Studio

Android Studio is designed to make testing simple. It contains many features to simplify how you create, run, and analyze tests. You can set up tests that run on your local machine or instrumented tests that run on a device. You can easily run a single test or a specific group of tests on one or more devices. The test results are shown directly inside Android Studio.

https://developer.android.com/studio/test/test-in-android-studio (last visited 5/1/2024).

Android Studio includes an emulator for testing mobile device applications. The emulator allows users to "test [their] application on a variety of devices" and "[i]n most cases, the emulator is the best option for your testing needs."



The Android Emulator simulates Android devices on your computer so that you can test your application on a variety of devices and Android API levels without needing to have each physical device. The emulator offers these advantages:

- **Flexibility**: In addition to being able to simulate a variety of devices and Android API levels, the emulator comes with predefined configurations for various Android phone, tablet, Wear OS, and Android TV devices.
- **High fidelity**: The emulator provides almost all the capabilities of a real Android device. You can simulate incoming phone calls and text messages, specify the location of the device, simulate different network speeds, simulate rotation and other hardware sensors, access the Google Play Store, and much more.
- **Speed**: Testing your app on the emulator is in some ways faster and easier than doing so on a physical device. For example, you can transfer data faster to the emulator than to a device connected over USB.

In most cases, the emulator is the best option for your testing needs. This page covers the core emulator functionalities and how to get started with it.

https://developer.android.com/studio/run/emulator (last visited 5/1/24).

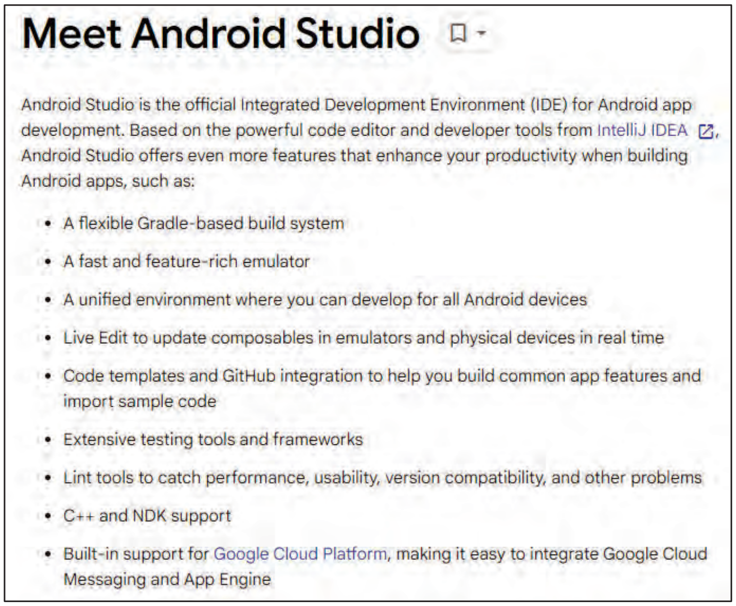
Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;

The Accused Instrumentalities include software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application.

Android Studio is software that is for testing applications based on the Android operating system. Testers use Android Studio to test applications by executing compiled source code for the application and monitoring the execution of the source code. Android Studio includes "[a] fast and feature-rich emulator" and "[e]xtensive testing tools and frameworks."



Meet Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development. Based on the powerful code editor and developer tools from IntelliJ IDEA, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

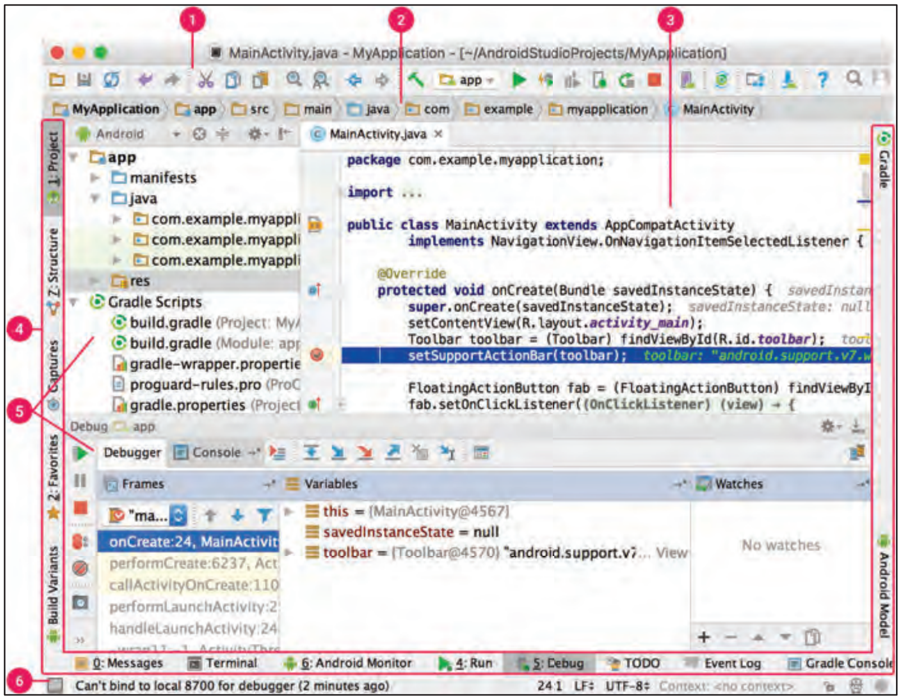
- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Live Edit to update composables in emulators and physical devices in real time
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

https://developer.android.com/studio/intro (last visited 4/6/2024).

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;

The following figure illustrates the Android Studio main window that is used for reviewing source code files.



```
package com.example.myapplication;

import ...

public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {  savedInstan
        super.onCreate(savedInstanceState);  savedInstanceState: null
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);  tool
        setSupportActionBar(toolbar);  toolbar: "android.support.v7.w

        FloatingActionButton fab = (FloatingActionButton) findViewByI
        fab.setOnClickListener((OnClickListener) (view) → {
```

```
this = {MainActivity@4567}
savedInstanceState = null
toolbar = {Toolbar@4570} "android.support.v7... View
```

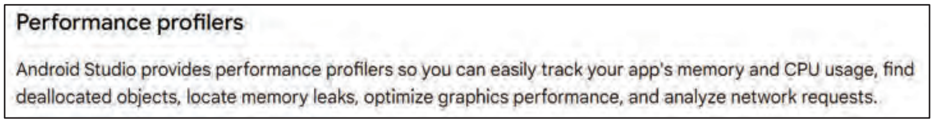
Can't bind to local 8700 for debugger (2 minutes ago)   24:1   LF   UTF-8   Context: <no context>

https://developer.android.com/studio/intro (last visited 4/6/2024). The editor window (#3) is used to review source code, which is part of software testing.

Android Studio's software interface includes a number of different "windows" or tools that are available to the application tester throughout the testing process. These windows include, for example, the performance profilers, heap dump, memory profiler, code inspection tools (e.g., Lint), Android Emulator, and Android Virtual Devices, each of which is described below.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

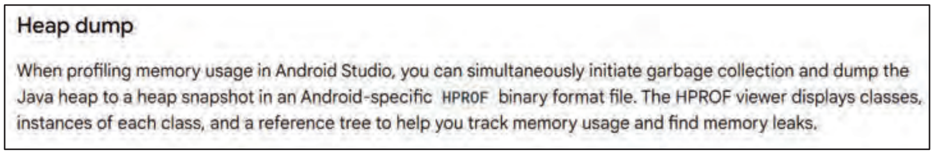
1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;



Performance profilers

Android Studio provides performance profilers so you can easily track your app's memory and CPU usage, find deallocated objects, locate memory leaks, optimize graphics performance, and analyze network requests.

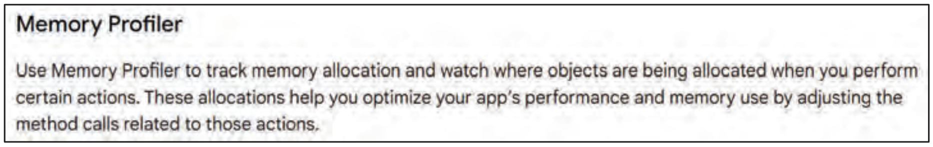
https://developer.android.com/studio/intro (last visited 5/1/2024).



Heap dump

When profiling memory usage in Android Studio, you can simultaneously initiate garbage collection and dump the Java heap to a heap snapshot in an Android-specific `HPROF` binary format file. The HPROF viewer displays classes, instances of each class, and a reference tree to help you track memory usage and find memory leaks.

https://developer.android.com/studio/intro (last visited 5/1/2024).



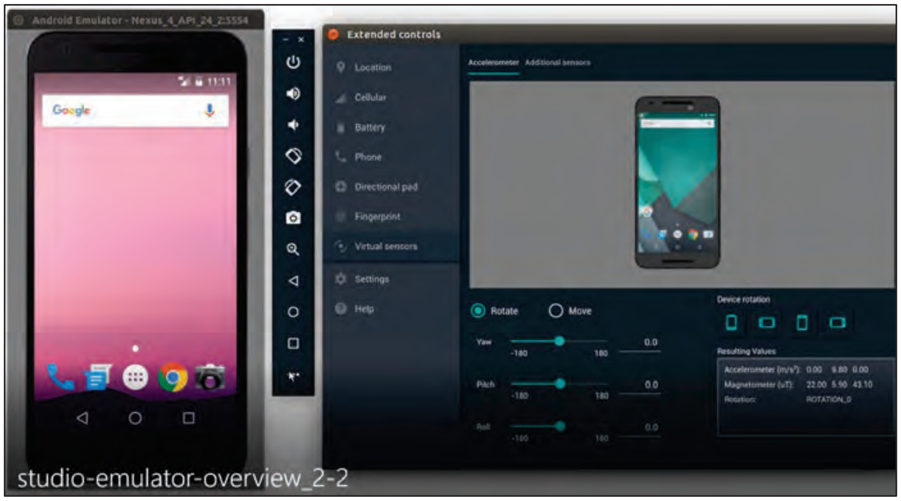
Memory Profiler

Use Memory Profiler to track memory allocation and watch where objects are being allocated when you perform certain actions. These allocations help you optimize your app's performance and memory use by adjusting the method calls related to those actions.

https://developer.android.com/studio/intro (last visited 5/1/2024).

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;



● Rotate ○ Move

Yaw -180 180 0.0

Pitch -180 180 0.0

Roll -180 180 0.0

Resulting Values

Accelerometer (m/s²): 0.00 9.80 0.00

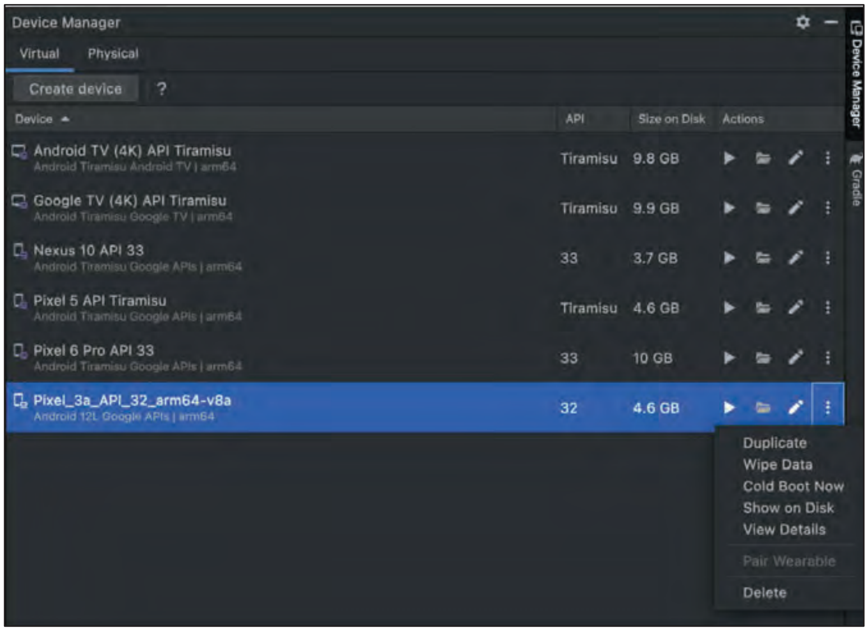
Magnetometer (uT): 22.00 5.90 43.10

Rotation: ROTATION_0

https://storage.googleapis.com/androiddevelopers/videos/studio-emulator-overview_2-2.mp4 (last visited 5/1/2024) (illustrating Android Emulator).

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;



Device	API	Size on Disk	Actions
Android TV (4K) API Tiramisu Android Tiramisu Android TV | arm64	Tiramisu	9.8 GB	
Google TV (4K) API Tiramisu Android Tiramisu Google TV | arm64	Tiramisu	9.9 GB	
Nexus 10 API 33 Android Tiramisu Google APIs | arm64	33	3.7 GB	
Pixel 5 API Tiramisu Android Tiramisu Google APIs | arm64	Tiramisu	4.6 GB	
Pixel 6 Pro API 33 Android Tiramisu Google APIs | arm64	33	10 GB	
Pixel_3a_API_32_arm64-v8a Android 12L Google APIs | arm64	32	4.6 GB	

https://developer.android.com/studio/run/managing-avds (last visited 5/1/2024) (illustrating Android Device Manager).

On information and belief, Defendant uses Android Studio while testing its mobile applications to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application being tested.

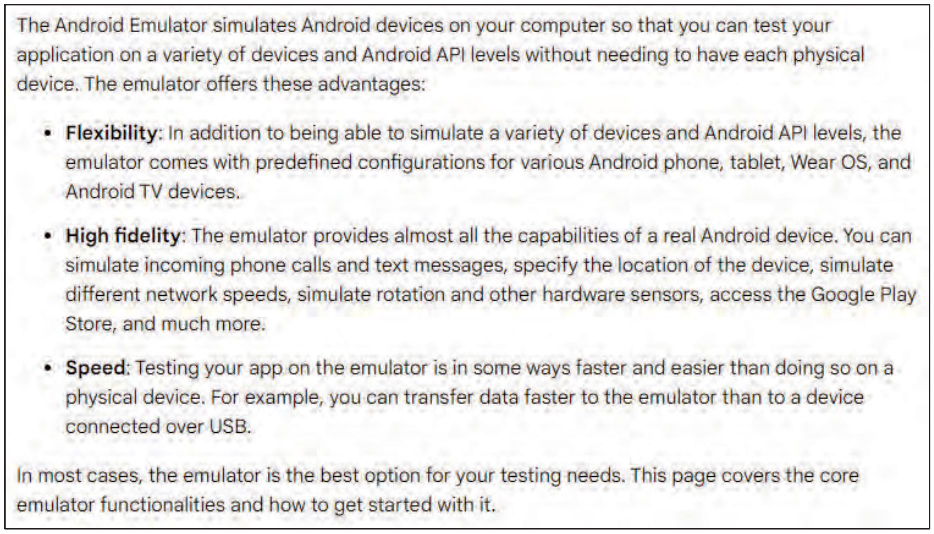
1. **Simulate a plurality of network characteristics**

Android Studio supports simulation of a plurality of network characteristics through the use of the Android Emulator and Android Virtual Devices (AVDs). The Android Emulator can be used to emulate/simulate Android devices on a computer without actually

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;

possessing the physical device. The Android Emulator emulates/simulates "almost all of the capabilities of a real Android device." Android Emulator includes predefined device configurations (or hardware profiles), and it also supports the use of customized device configurations that can be tailored to match the capabilities of a vast array of real-world Android devices.



The Android Emulator simulates Android devices on your computer so that you can test your application on a variety of devices and Android API levels without needing to have each physical device. The emulator offers these advantages:

- **Flexibility**: In addition to being able to simulate a variety of devices and Android API levels, the emulator comes with predefined configurations for various Android phone, tablet, Wear OS, and Android TV devices.
- **High fidelity**: The emulator provides almost all the capabilities of a real Android device. You can simulate incoming phone calls and text messages, specify the location of the device, simulate different network speeds, simulate rotation and other hardware sensors, access the Google Play Store, and much more.
- **Speed**: Testing your app on the emulator is in some ways faster and easier than doing so on a physical device. For example, you can transfer data faster to the emulator than to a device connected over USB.

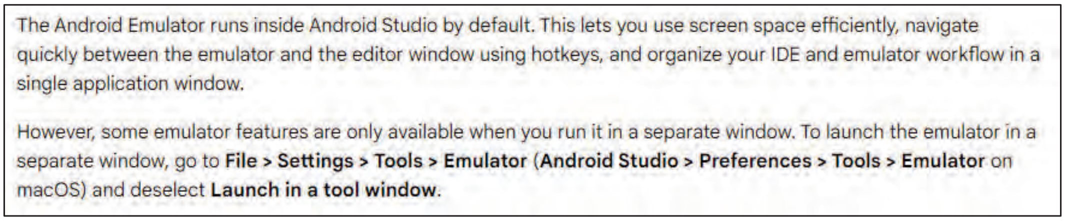
In most cases, the emulator is the best option for your testing needs. This page covers the core emulator functionalities and how to get started with it.

https://developer.android.com/studio/run/emulator (last visited 4/6/2024).

The Android Emulator is available directly within Android Studio and runs inside Android Studio by default. However, the emulator can also be launched in a separate tool window from within Android Studio as well.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;

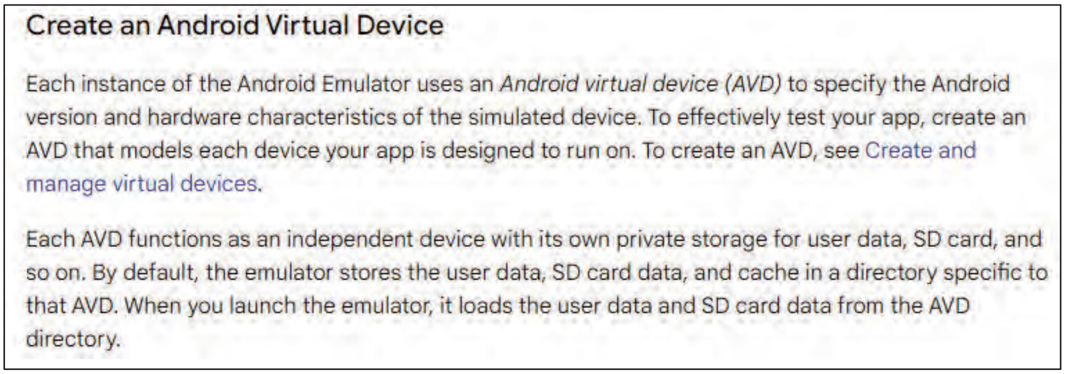


The Android Emulator runs inside Android Studio by default. This lets you use screen space efficiently, navigate quickly between the emulator and the editor window using hotkeys, and organize your IDE and emulator workflow in a single application window.

However, some emulator features are only available when you run it in a separate window. To launch the emulator in a separate window, go to **File > Settings > Tools > Emulator** (**Android Studio > Preferences > Tools > Emulator** on macOS) and deselect **Launch in a tool window**.

https://developer.android.com/studio/run/emulator-launch-separate-window (last visited 4/6/2024).

The device being emulated/simulated by Android Emulator is based on an Android Virtual Device (AVD). An AVD specifies the "hardware characteristics" of the device being emulated/simulated. Each AVD specifies the resources of the emulated/simulated device. These AVDs are created and managed in Android Studio using the Android Device Manager.[1]



Create an Android Virtual Device

Each instance of the Android Emulator uses an *Android virtual device (AVD)* to specify the Android version and hardware characteristics of the simulated device. To effectively test your app, create an AVD that models each device your app is designed to run on. To create an AVD, see Create and manage virtual devices.

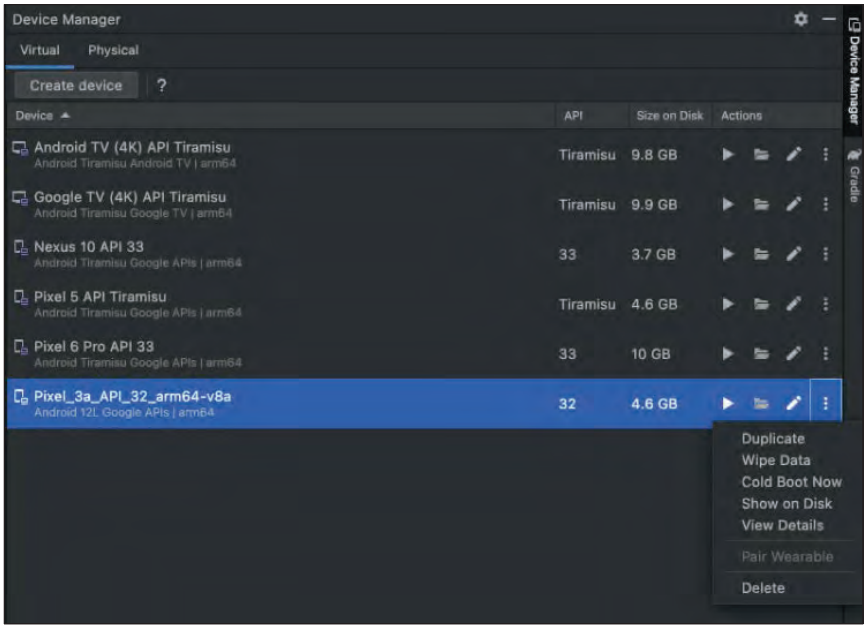
Each AVD functions as an independent device with its own private storage for user data, SD card, and so on. By default, the emulator stores the user data, SD card data, and cache in a directory specific to that AVD. When you launch the emulator, it loads the user data and SD card data from the AVD directory.

https://developer.android.com/studio/run/emulator (last visited 4/6/2024).

[1] Prior to the Bumblebee release, Device Manager was referred to as the Android Virtual Device Manager.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;



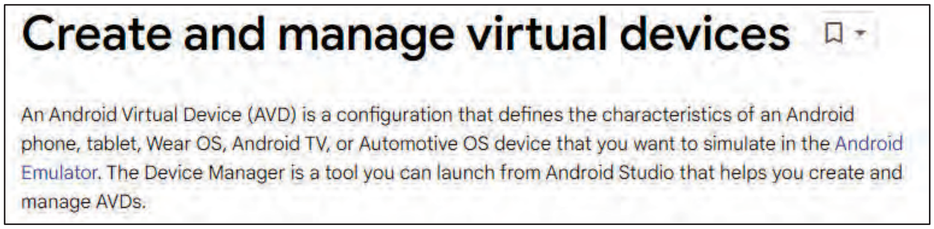
Device	API	Size on Disk	Actions
Android TV (4K) API Tiramisu Android Tiramisu Android TV | arm64	Tiramisu	9.8 GB	
Google TV (4K) API Tiramisu Android Tiramisu Google TV | arm64	Tiramisu	9.9 GB	
Nexus 10 API 33 Android Tiramisu Google APIs | arm64	33	3.7 GB	
Pixel 5 API Tiramisu Android Tiramisu Google APIs | arm64	Tiramisu	4.6 GB	
Pixel 6 Pro API 33 Android Tiramisu Google APIs | arm64	33	10 GB	
Pixel_3a_API_32_arm64-v8a Android 12L Google APIs | arm64	32	4.6 GB	

https://developer.android.com/studio/run/managing-avds (last visited 4/6/2024) (illustrating Android Device Manager).

Once an Android Virtual Device is created, its operating properties can be specified. All of these properties are simulated/ emulated during operation within the Emulator.

<u>Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)</u>

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;



Create and manage virtual devices

An Android Virtual Device (AVD) is a configuration that defines the characteristics of an Android phone, tablet, Wear OS, Android TV, or Automotive OS device that you want to simulate in the Android Emulator. The Device Manager is a tool you can launch from Android Studio that helps you create and manage AVDs.

https://developer.android.com/studio/run/managing-avds (last visited 4/6/2024).

Each Android Virtual Device has a profile that includes a number of properties defining the characteristics of the AVD to emulate/simulate. The "hardware profile properties" include:

- Device Name
- Device Type
- Screen: Screen Size
- Screen: Screen Resolution
- Screen: Round
- Memory: RAM
- Input: Has Hardware Buttons (Back/Home/Menu)
- Input: Has Hardware Keyboard
- Input: Navigation Style
- Supported Device States
- Cameras
- Sensors: Accelerometer
- Sensors: Gyroscope
- Sensors: GPS
- Sensors: Proximity Sensor
- Default Skin.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;

See https://developer.android.com/studio/run/managing-avds (last visited 4/6/2024) (listing and describing hardware profile properties). Additionally, an AVD has AVD properties that also control the manner in which the AVD performs during emulation/simulation. The AVD Properties include:

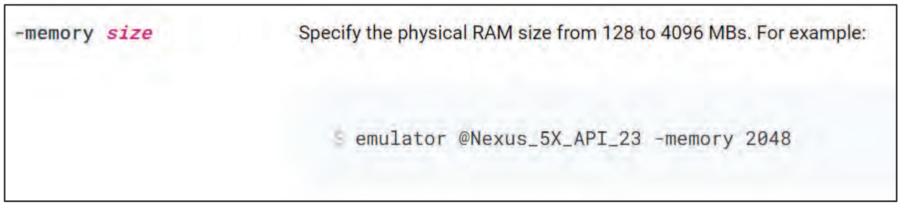
- AVD Name
- AVD ID (Advanced)
- Hardware Profile
- System Image
- Startup Orientation
- Camera (Advanced)
- Network: Speed (Advanced)
- Network: Latency (Advanced)
- Emulated Performance: Graphics
- Emulated Performance: Boot option (Advanced)
- Emulated Performance: Multi-Core CPU (Advanced)
- Memory and Storage: RAM (Advanced)
- Memory and Storage: VM Heap (Advanced)
- Memory and Storage: Internal Storage (Advanced)
- Memory and Storage: SD Card (Advanced)
- Device Frame: Enable Device Frame
- Custom Skin Definition (Advanced)
- Keyboard: Enable Keyboard Input (Advanced)

See https://developer.android.com/studio/run/managing-avds (last visited 4/6/2024) (listing and describing AVD properties). Each of the above-mentioned properties are emulated/simulated by Android Emulator when running the device profile specified for that particular AVD. The hardware profile properties and the AVD properties represent device characteristics, including hardware characteristics and network characteristics. When the Emulator runs a particular AVD, these characteristics are emulated/simulated and are indicative of the performance of the emulated/simulated device when testing the application.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;

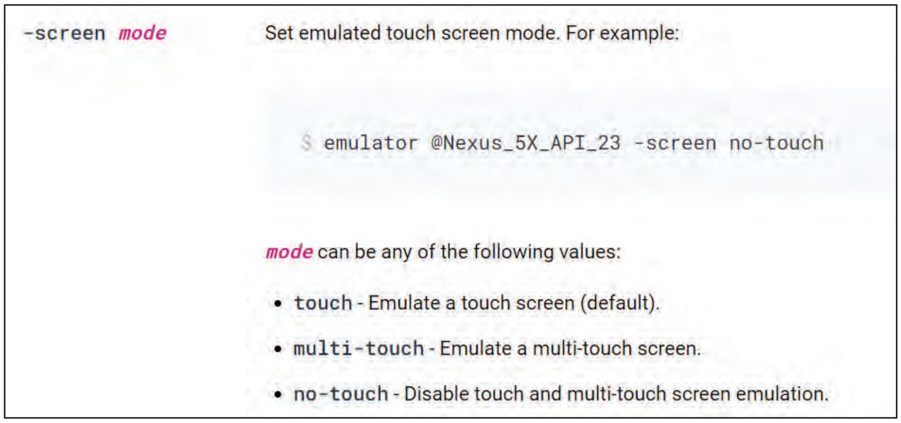
Some hardware profile properties and AVD properties can also be controlled via the command-line options available for the Android Emulator. Examples of the command-line options permitting control of hardware characteristics of the AVD device during simulation are provided below.



`-memory size` Specify the physical RAM size from 128 to 4096 MBs. For example:

```
$ emulator @Nexus_5X_API_23 -memory 2048
```

https://developer.android.com/studio/run/emulator-commandline (last visited 4/6/2024) (showing options for controlling the RAM size of the emulated/simulated device).



`-screen mode` Set emulated touch screen mode. For example:

```
$ emulator @Nexus_5X_API_23 -screen no-touch
```

mode can be any of the following values:

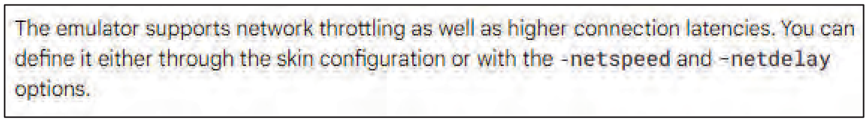
- `touch` - Emulate a touch screen (default).
- `multi-touch` - Emulate a multi-touch screen.
- `no-touch` - Disable touch and multi-touch screen emulation.

https://developer.android.com/studio/run/emulator-commandline (last visited 4/6/2024) (showing options for controlling the touch capabilities to emulate/simulate for the device screen).

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;

The Android Emulator is programmed to emulate a plurality of network characteristics, such as network speed (upload/download) and network latency.



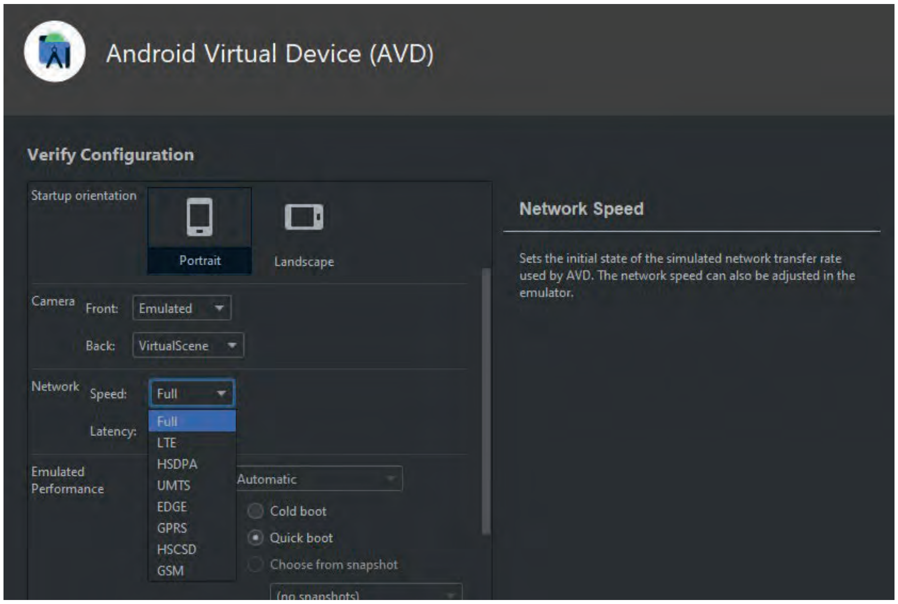
The emulator supports network throttling as well as higher connection latencies. You can define it either through the skin configuration or with the `-netspeed` and `-netdelay` options.

https://developer.android.com/studio/run/emulator-commandline (last visited 4/6/2024).

The following figures show screenshots of the configuration screens for an Android Virtual Device. The first figure shows how an AVD can have a specific network speed associated with it, the speed options including: Full, LTE, HSDPA, UMTS, EDGE, GPRS, HSCSD, GSM.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;



Android Virtual Device (AVD)

Verify Configuration

Startup orientation: Portrait

Camera Front: Emulated

Back: VirtualScene

Network Speed: Full

Network Speed

Sets the initial state of the simulated network transfer rate used by AVD. The network speed can also be adjusted in the emulator.

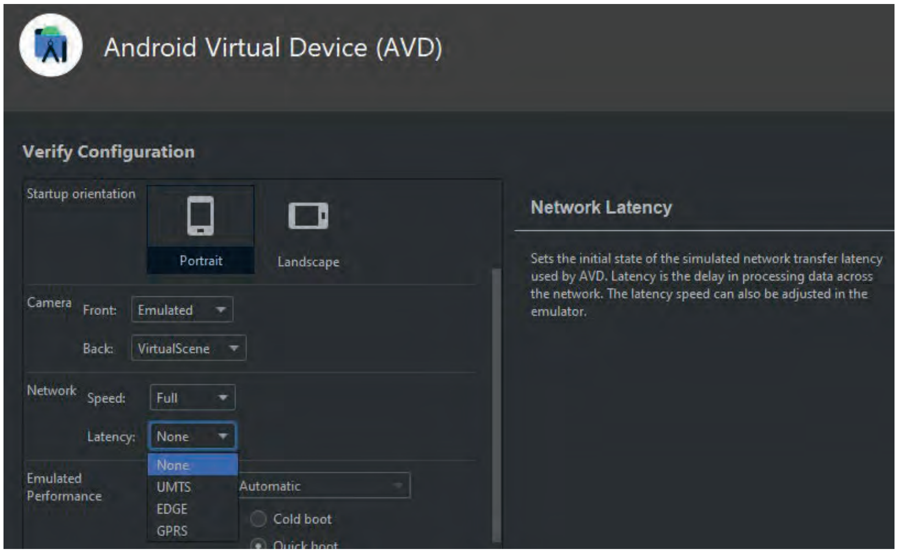
Screenshot from Android Studio Arctic Fox[2]: Android Virtual Device (showing Network Speed options: Full, LTE, HSDPA, UMTS, EDGE, GPRS, HSCSD, GSM).

The following screenshot shows how an AVD can have a specific network latency associated with it, the latency options including, for example: None, UMTS, EDGE, GPRS.

[2] New versions of Android Studio are released on a regular basis. The user interface may differ slightly between different versions; however, the functionality identified in this chart exists in Android Studio versions from 2017 to the present. To the extent the functionality in different versions of Android Studio changes in a meaningful way regarding the infringement read, those changes will be noted in the chart.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;



Android Virtual Device (AVD)

Verify Configuration

Startup orientation: Portrait

Camera Front: Emulated

Back: VirtualScene

Network Speed: Full

Latency: None

Emulated Performance: Automatic

Network Latency

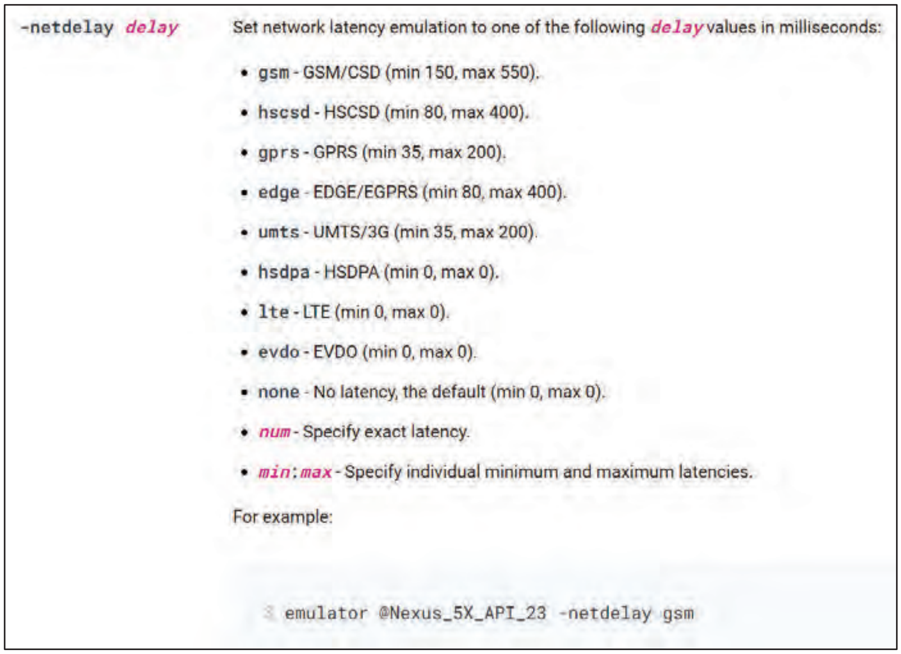
Sets the initial state of the simulated network transfer latency used by AVD. Latency is the delay in processing data across the network. The latency speed can also be adjusted in the emulator.

Screenshot from Android Studio Arctic Fox: Android Virtual Device (showing Network Latency options).

The network characteristics can also be controlled via the command-line options available for the Android Emulator, as illustrated below.

<u>Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)</u>

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;



`-netdelay delay` Set network latency emulation to one of the following `delay` values in milliseconds:

- `gsm` - GSM/CSD (min 150, max 550).
- `hscsd` - HSCSD (min 80, max 400).
- `gprs` - GPRS (min 35, max 200).
- `edge` - EDGE/EGPRS (min 80, max 400).
- `umts` - UMTS/3G (min 35, max 200).
- `hsdpa` - HSDPA (min 0, max 0).
- `lte` - LTE (min 0, max 0).
- `evdo` - EVDO (min 0, max 0).
- none - No latency, the default (min 0, max 0).
- *`num`* - Specify exact latency.
- *`min`*`:`*`max`* - Specify individual minimum and maximum latencies.

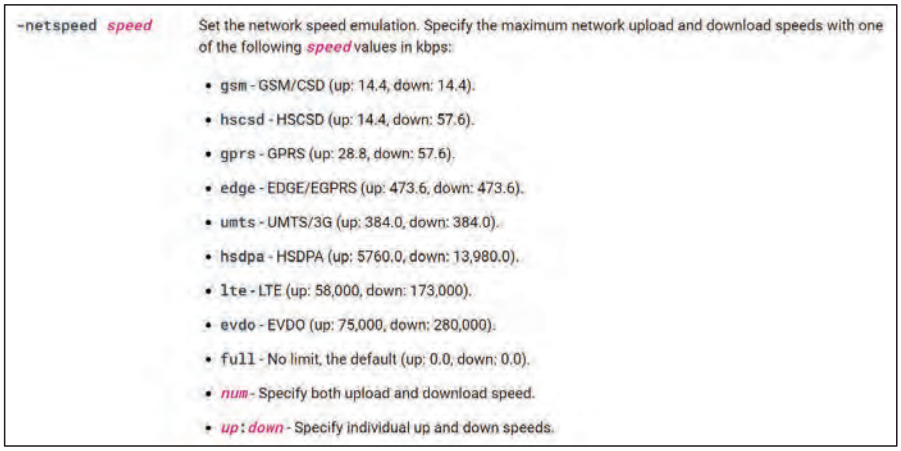
For example:

```
$ emulator @Nexus_5X_API_23 -netdelay gsm
```

https://developer.android.com/studio/run/emulator-commandline (last visited 4/7/2024) (showing options for setting network latency properties).

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;



-netspeed *speed*	Set the network speed emulation. Specify the maximum network upload and download speeds with one of the following *speed* values in kbps: - gsm - GSM/CSD (up: 14.4, down: 14.4). - hscsd - HSCSD (up: 14.4, down: 57.6). - gprs - GPRS (up: 28.8, down: 57.6). - edge - EDGE/EGPRS (up: 473.6, down: 473.6). - umts - UMTS/3G (up: 384.0, down: 384.0). - hsdpa - HSDPA (up: 5760.0, down: 13,980.0). - lte - LTE (up: 58,000, down: 173,000). - evdo - EVDO (up: 75,000, down: 280,000). - full - No limit, the default (up: 0.0, down: 0.0). - *num* - Specify both upload and download speed. - *up*:*down* - Specify individual up and down speeds.

https://developer.android.com/studio/run/emulator-commandline (last visited 4/7/2024) (showing options for setting network speed properties).

These network characteristics (e.g., speed and latency) are indicative of the performance of a mobile device, affecting the speed and latency with which it can communicate over a network connection while executing the application.

2. Profile display windows

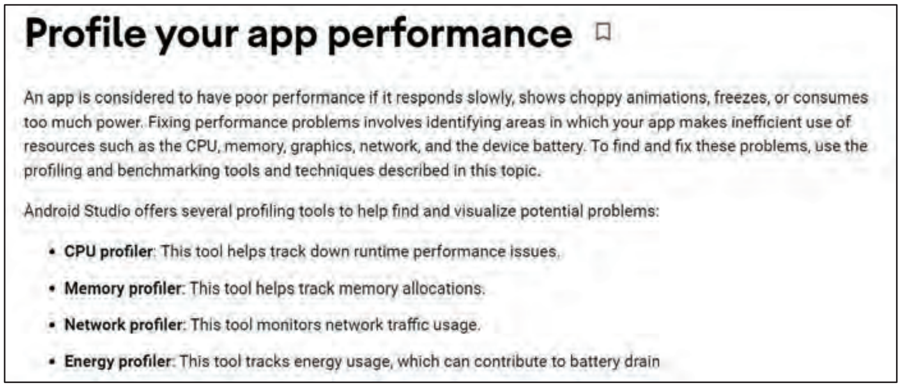
Android Studio includes a number of profiling tools that support application testing, allowing a software tester to monitor the resources of the Android device or AVD that are used by and available to the application while executing on the device. These profiling tools have corresponding display windows.

The Android profiling tools for Arctic Fox and relevant prior releases include: (1) CPU Profiler; (2) Memory Profiler; (3) Network Profiler; and (4) Energy Profiler. Starting with the Bumblebee release, the Network Profiler was moved to App Inspection and renamed the Network Inspector, as detailed further below.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;

These profiling tools are used for "[f]ixing performance problems [which] involves identifying areas in which your app makes inefficient use of resources such as the CPU, memory, graphics, network, and the device battery . . . . Android studio offers several profiling tools to help find and visualize potential problems."



Profile your app performance

An app is considered to have poor performance if it responds slowly, shows choppy animations, freezes, or consumes too much power. Fixing performance problems involves identifying areas in which your app makes inefficient use of resources such as the CPU, memory, graphics, network, and the device battery. To find and fix these problems, use the profiling and benchmarking tools and techniques described in this topic.

Android Studio offers several profiling tools to help find and visualize potential problems:

- **CPU profiler**: This tool helps track down runtime performance issues.
- **Memory profiler**: This tool helps track memory allocations.
- **Network profiler**: This tool monitors network traffic usage.
- **Energy profiler**: This tool tracks energy usage, which can contribute to battery drain

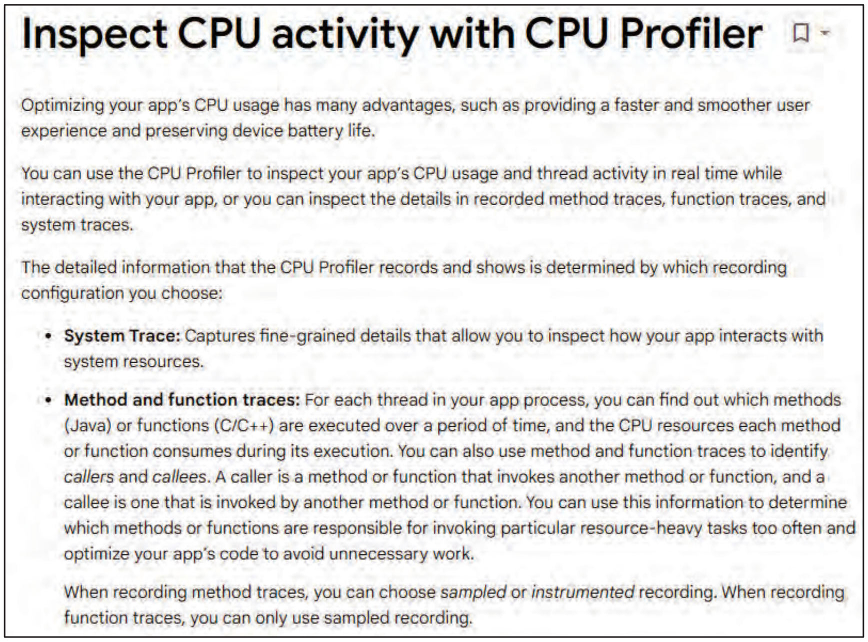
https://developer.android.com/studio/profile (last visited 1/6/2022).

Android Studio allows the application tester to profile the CPU, memory usage, network activity, and energy use of the mobile device while executing and testing an application. Each profiler is detailed below.

The **CPU Profiler** is used to monitor CPU usage and availability, and it helps track down runtime performance issues. It can be used to "inspect your app's CPU usage and thread activity in real time while interacting with your app . . . ."

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;



Inspect CPU activity with CPU Profiler

Optimizing your app's CPU usage has many advantages, such as providing a faster and smoother user experience and preserving device battery life.

You can use the CPU Profiler to inspect your app's CPU usage and thread activity in real time while interacting with your app, or you can inspect the details in recorded method traces, function traces, and system traces.

The detailed information that the CPU Profiler records and shows is determined by which recording configuration you choose:

- **System Trace:** Captures fine-grained details that allow you to inspect how your app interacts with system resources.
- **Method and function traces:** For each thread in your app process, you can find out which methods (Java) or functions (C/C++) are executed over a period of time, and the CPU resources each method or function consumes during its execution. You can also use method and function traces to identify *callers* and *callees*. A caller is a method or function that invokes another method or function, and a callee is one that is invoked by another method or function. You can use this information to determine which methods or functions are responsible for invoking particular resource-heavy tasks too often and optimize your app's code to avoid unnecessary work.

  When recording method traces, you can choose *sampled* or *instrumented* recording. When recording function traces, you can only use sampled recording.

https://developer.android.com/studio/profile/cpu-profiler (last visited 4/7/2024) (detailing the CPU Profiler). The CPU Profiler displays the executed application's CPU usage, CPU availability, and thread activity via timelines, as illustrated below.

<u>Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)</u>

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;

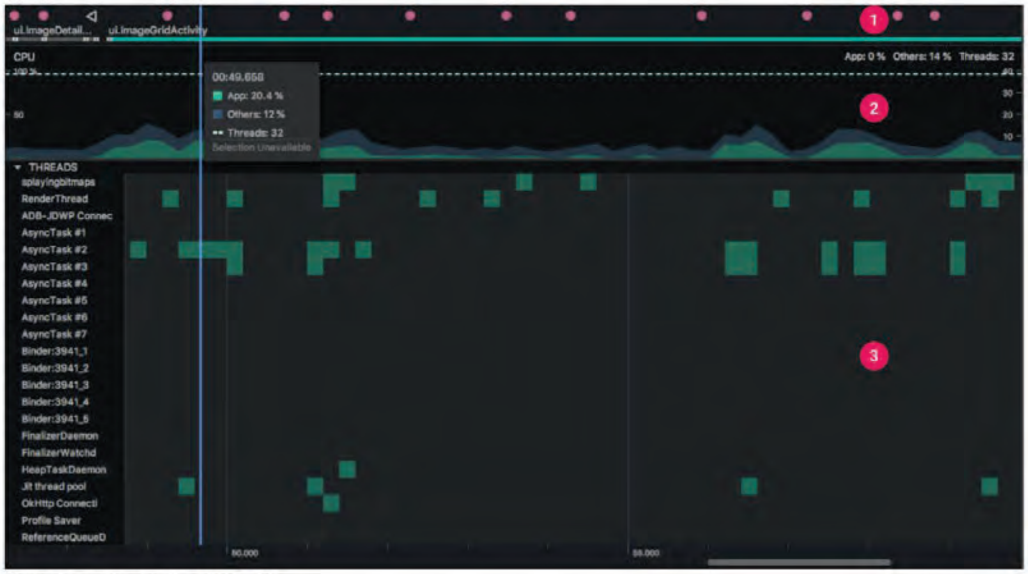


Figure 1. Timelines in the CPU Profiler.

https://developer.android.com/studio/profile/cpu-profiler (last visited 4/7/2024) (illustrating the CPU Profiler timelines).

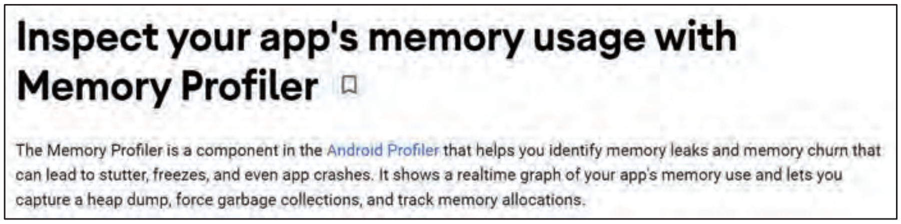
Referring to the numbers in the figure above, Number 1 illustrates the Event Timeline, which "[s]hows the activities in your app as they transition through different states in their lifecycle, and indicates user interactions with the device, including screen rotation events." *Id.* Number 2 illustrates the CPU Timeline, which "[s]hows real-time CPU usage of your app—as a percentage of total available CPU time—and the total number of threads your app is using. The timeline also shows the CPU usage of other processes (such as system processes or other apps), so you can compare it to your app's usage. You can inspect historical CPU usage data by moving your mouse along the horizontal axis of the timeline." *Id.* Finally, Number 3 indicates the Thread Activity Timeline, which "[l]ists each thread that belongs to your app process and indicates its activity along a timeline using the colors listed below." *Id.*

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;

The CPU Timeline (Number 2) shows the CPU usage of the application and CPU availability during execution. In particular, the lighter green portions of the CPU timeline illustrate the percentage of CPU resources used by the application at any given point in time. The darker green/gray portion of the timeline shows the CPU resources consumed by other components on the device (e.g., system processes or other applications). Finally, the remaining dark/black portion of the timeline shows the amount of CPU resources available to the application. The dotted white line across the top of the CPU Timeline shows the 100% mark, indicating the maximum CPU resources available for consumption.

The **Memory Profiler** is used to monitor memory usage by the application as well as memory available to the application. It assists with detecting unwanted or unnecessary memory consumption, including memory leaks and memory churn.



Inspect your app's memory usage with Memory Profiler

The Memory Profiler is a component in the Android Profiler that helps you identify memory leaks and memory churn that can lead to stutter, freezes, and even app crashes. It shows a realtime graph of your app's memory use and lets you capture a heap dump, force garbage collections, and track memory allocations.

https://developer.android.com/studio/profile/memory-profiler (last visited 2/7/2024) (describing the Memory Profiler). An example view of the Memory Profiler is provided below:

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;

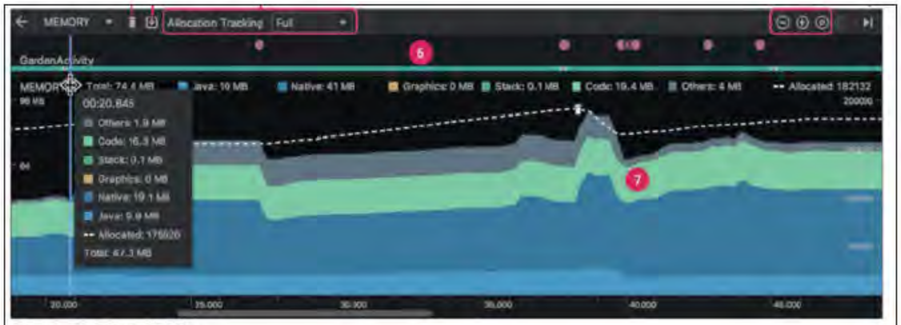


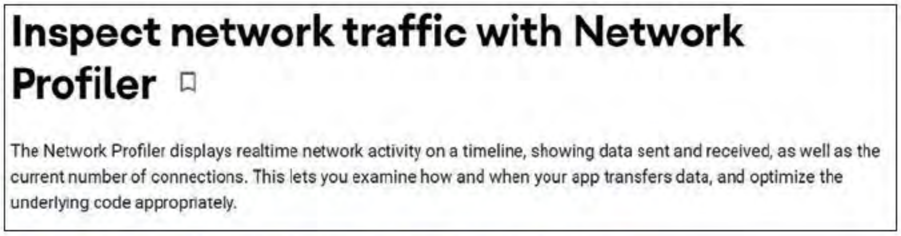
Figure 1. The Memory Profiler

https://developer.android.com/studio/profile/memory-profiler (last visited 4/7/2024) (illustrating the Memory Profiler). The memory legend near the top illustrates the amount of memory consumed or utilized by the application (e.g., Total, Java, Native, Graphics, Stack, Code, Others). Based on the AVD memory limit, this also indicates the total memory resources (unallocated and/or allocated) available to the application (i.e., hardware profile memory available minus total consumed). In addition, the Allocated component (represented in the graph by a white dotted line) indicates the amount of allocated memory resources currently available to the application. Further, the AVD includes a limit on the amount of memory (unallocated and/or allocated) available to the application while being simulated/emulated on the AVD device. In this way, the Memory Profiler provides information about the unallocated and allocated memory resources available to and utilized by an application for a given AVD configuration.

The **Network Profiler** allows the application author to monitor network connections and data exchanges by the mobile application.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;

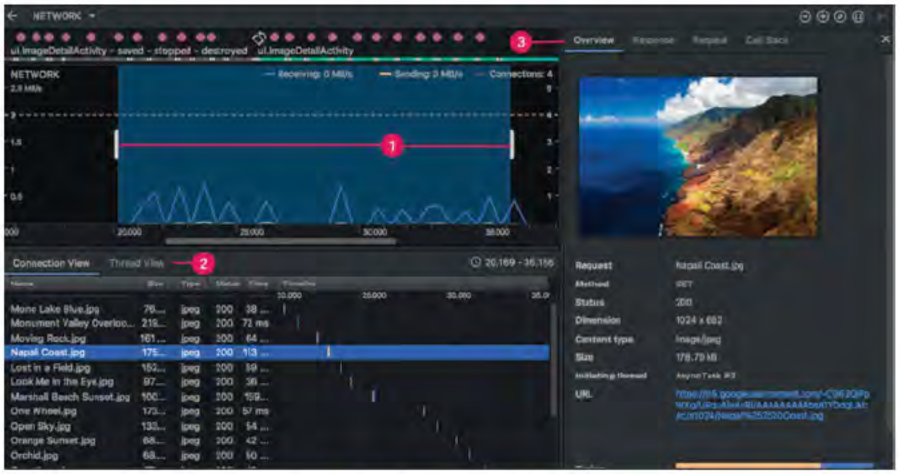


Inspect network traffic with Network Profiler

The Network Profiler displays realtime network activity on a timeline, showing data sent and received, as well as the current number of connections. This lets you examine how and when your app transfers data, and optimize the underlying code appropriately.

https://developer.android.com/studio/profile/network-profiler (last visited 1/6/2022).

As illustrated below, the Network Profiler shows the receiving network speed, the sending network speed, and latency.



Name	Size	Type	Status	Time	Timeline
Mono Lake Blue.jpg	76....	jpeg	200	38 ...	
Monument Valley Overloo...	219...	jpeg	200	72 ms	
Moving Rock.jpg	161 ...	jpeg	200	64 ...	
Napali Coast.jpg	175...	jpeg	200	113 ...	
Lost in a Field.jpg	152...	jpeg	200	59 ...	
Look Me in the Eye.jpg	97....	jpeg	200	38 ...	
Marshall Beach Sunset.jpg	100...	jpeg	200	159...	
One Wheel.jpg	173...	jpeg	200	57 ms	
Open Sky.jpg	133...	jpeg	200	54 ...	
Orange Sunset.jpg	68....	jpeg	200	42 ...	
Orchid.jpg	68....	jpeg	200	60 ...	

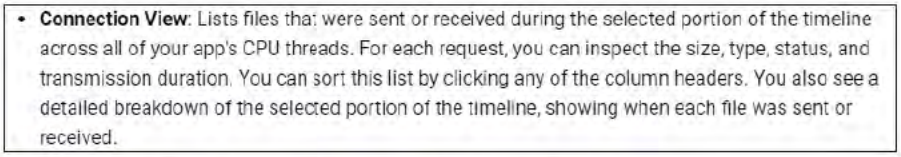
Request	Napali Coast.jpg
Method	GET
Status	200
Dimension	1024 x 682
Content type	image/jpeg
Size	178.79 kB
Initiating thread	AsyncTask #3
URL	[illegible]

https://developer.android.com/studio/profile/network-profiler (last visited 7/27/2021).

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;

The Connection View provides additional information about the network transmissions, including transmission duration and timing, which is related to network latency.



- **Connection View**: Lists files tha: were sent or received during the selected portion of the timeline across all of your app's CPU threads. For each request, you can inspect the size, type, status, and transmission duration. You can sort this list by clicking any of the column headers. You also see a detailed breakdown of the selected portion of the timeline, showing when each file was sent or received.

https://developer.android.com/studio/profile/network-profiler (last visited 7/27/2021).

The Thread View displays the network activity for each of the application's CPU threads, illustrating the receiving network speed, the sending network speed, and latency.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;

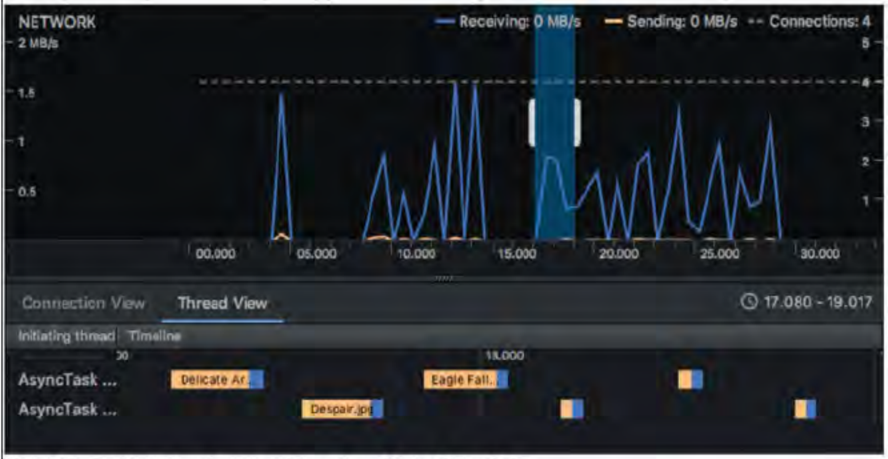


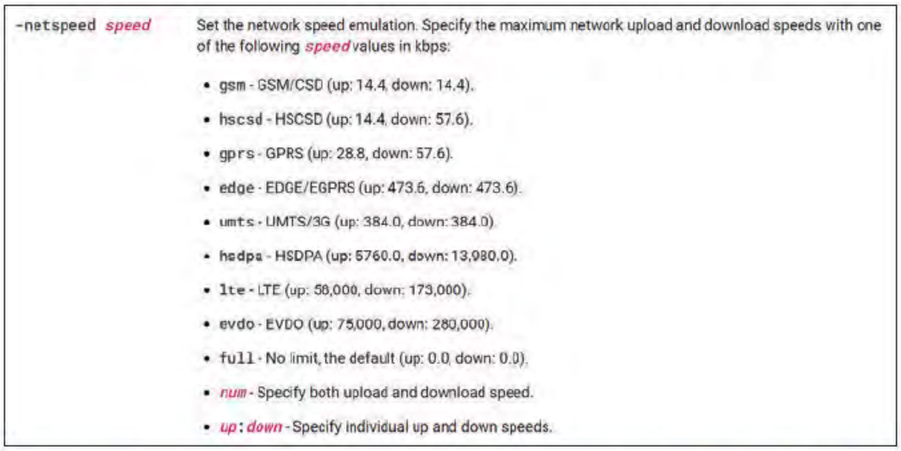
Figure 2. Inspecting network requests by app thread in the **Thread View**

https://developer.android.com/studio/profile/network-profiler (last visited 1/7/2022). The blue line in the network timeline shows the receiving rate of data flow, and the orange line shows the sending rate of data flow for the application. Each of these translate to bandwidth resources used by the application. The dotted line indicates the number of connections and the maximum data transfer rate employed by the application. The indication of a previously seen maximum data rate provides information about the bandwidth resources available to the application.

Additionally, as detailed above, the AVD is configured with network throttling properties that limit the bandwidth—both upload and download speeds—that is available to the application.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;



`-netspeed` *speed* Set the network speed emulation. Specify the maximum network upload and download speeds with one of the following *speed* values in kbps:

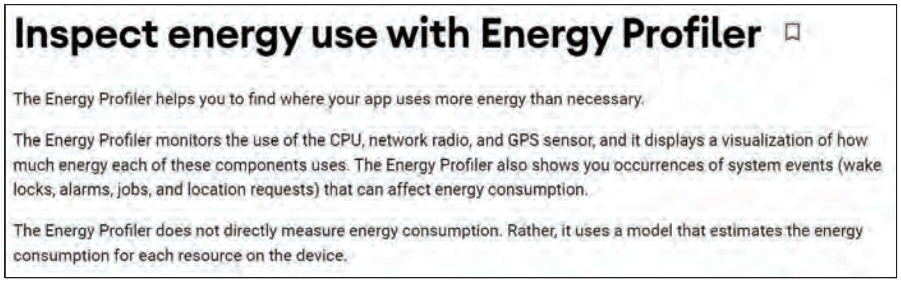
- `gsm` - GSM/CSD (up: 14.4, down: 14.4).
- `hscsd` - HSCSD (up: 14.4, down: 57.6).
- `gprs` - GPRS (up: 28.8, down: 57.6).
- `edge` - EDGE/EGPRS (up: 473.6, down: 473.6).
- `umts` - UMTS/3G (up: 384.0, down: 384.0).
- `hsdpa` - HSDPA (up: 5760.0, down: 13,980.0).
- `lte` - LTE (up: 58,000, down: 173,000).
- `evdo` - EVDO (up: 75,000, down: 280,000).
- `full` - No limit, the default (up: 0.0, down: 0.0).
- *num* - Specify both upload and download speed.
- *up*`:`*down* - Specify individual up and down speeds.

https://developer.android.com/studio/run/emulator-commandline (last visited 4/6/2024) (showing the upload and download maximums for different "network speed" settings provided in the AVD configuration). For example, an AVD configured with EDGE network speed is limited to 473.6 kbps upload speed and 473.6 kbps download speed. This provides an upper-bound on the network bandwidth resources available to the application, and the Network Profiler displays the utilized bandwidth per unit time, thus showing the remaining bandwidth available to the application based on its network speed settings.

The **Energy Profiler** allows the author to monitor energy consumption by the application.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;



Inspect energy use with Energy Profiler

The Energy Profiler helps you to find where your app uses more energy than necessary.

The Energy Profiler monitors the use of the CPU, network radio, and GPS sensor, and it displays a visualization of how much energy each of these components uses. The Energy Profiler also shows you occurrences of system events (wake locks, alarms, jobs, and location requests) that can affect energy consumption.

The Energy Profiler does not directly measure energy consumption. Rather, it uses a model that estimates the energy consumption for each resource on the device.

https://developer.android.com/studio/profile/energy-profiler (last visited 4/7/2024). An example view of the Energy Profiler is provided below:

<u>Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)</u>

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;

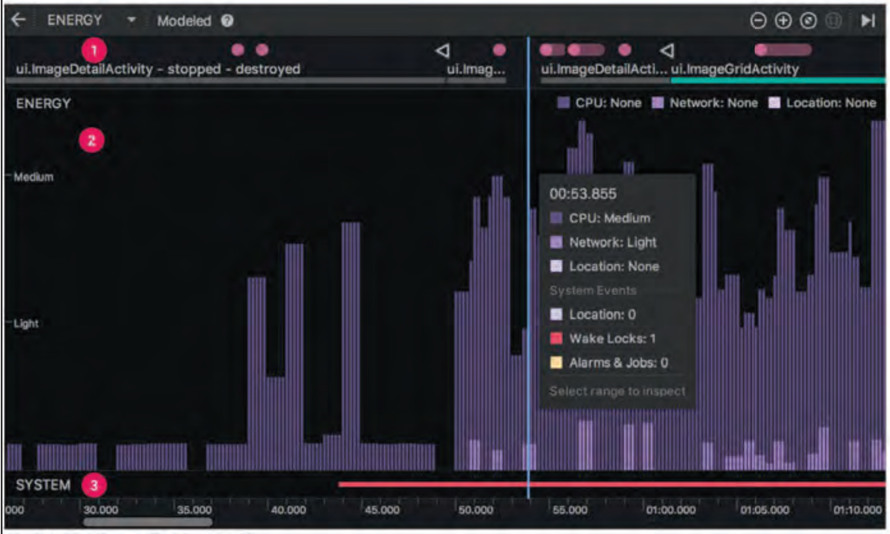


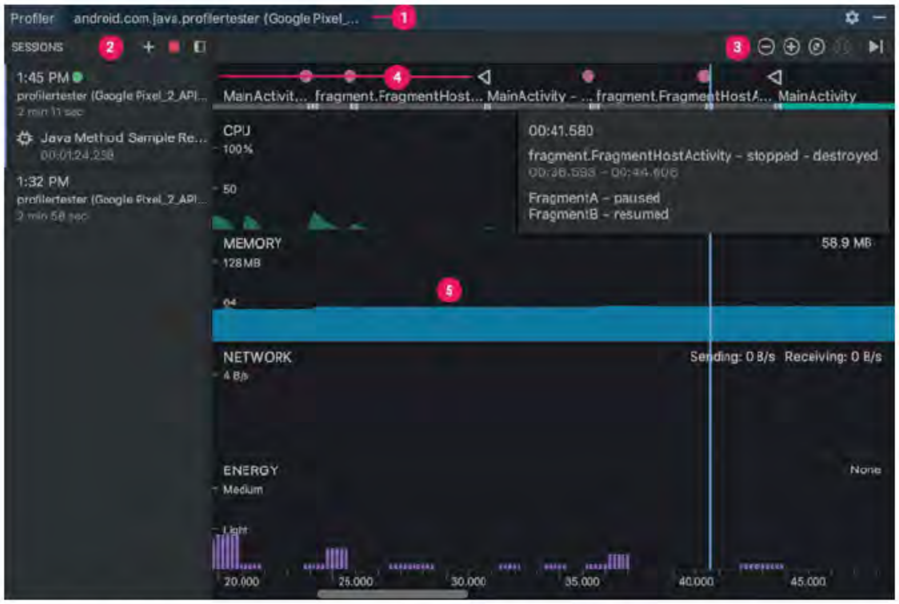
Figure 1. Timelines in the Energy Profiler.

https://developer.android.com/studio/profile/energy-profiler (last visited 4/7/2024). Three timelines are illustrated in this Energy Profiler view. The first (1) is the Event Timeline, which "[s]hows the activities in your app as they transition through different states in their lifecycle. This timeline also indicates user interactions with the device, including screen rotation events." *Id.* The second (2) is the Energy Timeline, which "[s]hows estimated energy consumption of your app." *Id.* And the third (3) is the System Time, which "[i]ndicates system events that may affect energy consumption." *Id.* By moving your mouse over the timelines, you can "see a breakdown of energy use by CPU, network, and location (GPS) resources . . . ." *Id.*

In addition to the exemplary profile display windows illustrated above, Android Studio Arctic Fox (and earlier versions) supports display of profile windows for all four profilers discussed above:

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;

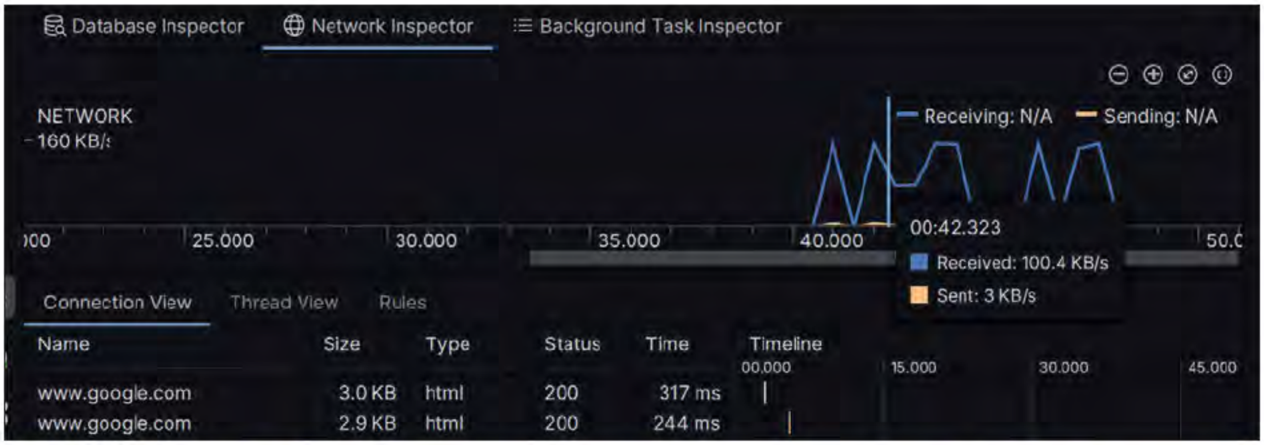


https://developer.android.com/studio/profile/android-profiler (last visited 7/27/2021) (illustrating the CPU, Memory, Network, and Energy profile displays).

Starting with the Android Studio Bumblebee release, the Network Profiler was moved to the App Inspection component of Android Studio and renamed Network Inspector. *See* https://developer.android.com/studio/releases/past-releases/as-bumblebee-release-notes (last visited 4/20/2024). The Network Inspector view is shown below:

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[B] software configured to simulate, via one or more profile display windows, a plurality of network characteristics indicative of performance of the mobile device when executing the application;



Name	Size	Type	Status	Time	Timeline
www.google.com	3.0 KB	html	200	317 ms	
www.google.com	2.9 KB	html	200	244 ms	

Screenshot from Android Studio Iguana (showing Network Inspector). The Network Inspector view shows the receiving network speed and the sending network speed. And it includes the Connection View and Thread View discussed above for the Network Profiler.

The CPU and Memory Profilers are available in post-Arctic Fox versions of Android Studio, as detailed above for Arctic Fox.

1[C] wherein the network characteristics are based on data of interaction with networks in non-simulated environments.

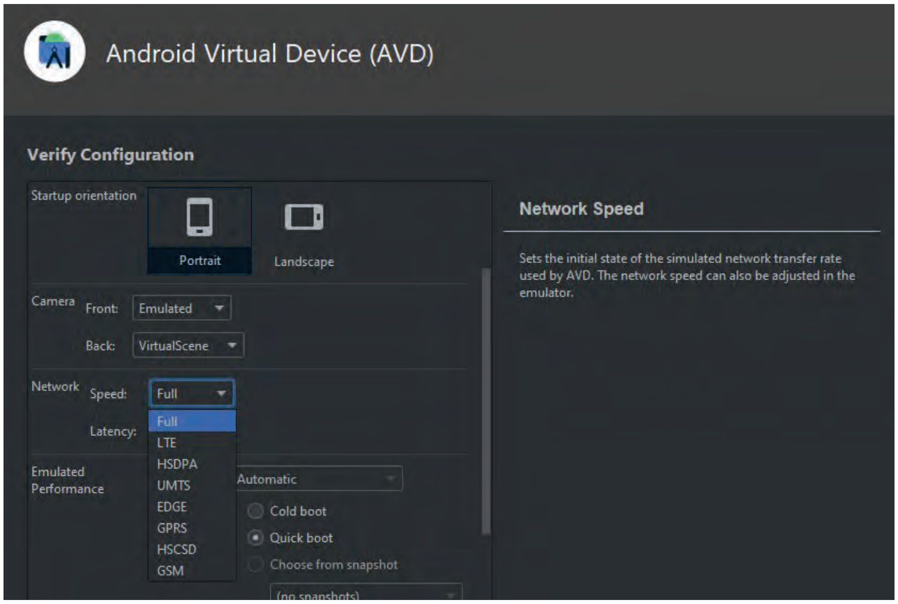
1[C] wherein the network characteristics are based on data of interaction with networks in non-simulated environments.

The network characteristics emulated/simulated by Android Emulator within Android Studio are based on data of interaction with networks in non-simulated environments. *See* https://developer.android.com/studio/run/emulator-console (last visited 5/1/2024) ("The emulator lets you simulate various network latency levels ***so that you can test your app in an environment more typical of actual running conditions***. You can set a latency level or range at emulator startup, or you can use the console to change the latency while the app is running in the emulator." (emphasis added)).

The network speed and network latency options for AVDs are based on data of interaction with networks in non-simulated environments, or based on real-world networks. For example, the Network speed options are based on real networks, including LTE, HSDPA, UMTS, EDGE, GPRS, HSCSD, and GSM networks.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Study Tools)

1[C] wherein the network characteristics are based on data of interaction with networks in non-simulated environments.



Android Virtual Device (AVD)

Verify Configuration

Startup orientation: Portrait

Camera Front: Emulated

Back: VirtualScene

Network Speed: Full

Network Speed

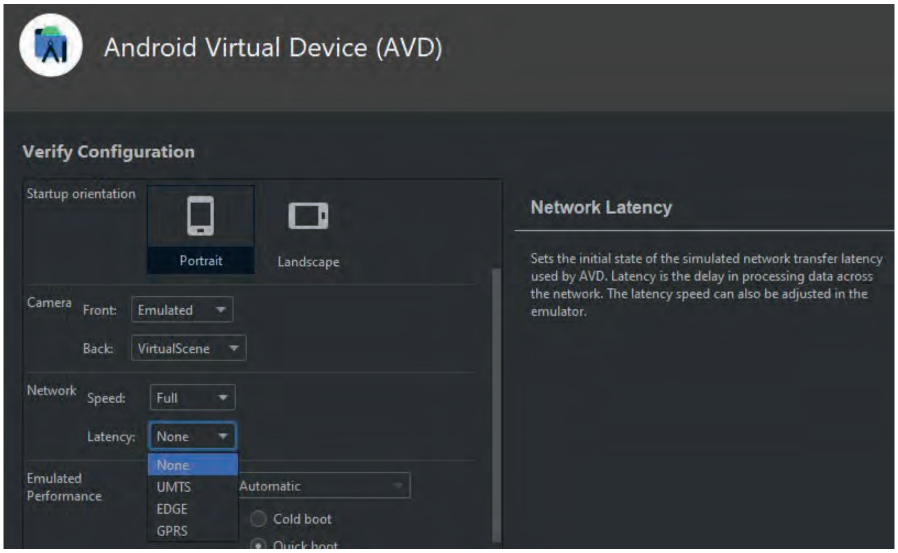
Sets the initial state of the simulated network transfer rate used by AVD. The network speed can also be adjusted in the emulator.

Screenshot from Android Studio Arctic Fox: Android Virtual Device.

Network latency options are similarly based on real-world "non-simulated" network information, including UMTS, EDGE, and GPRS networks.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[C] wherein the network characteristics are based on data of interaction with networks in non-simulated environments.



Android Virtual Device (AVD)

Verify Configuration

Startup orientation: Portrait

Camera Front: Emulated

Back: VirtualScene

Network Speed: Full

Latency: None

Emulated Performance: Automatic

Network Latency

Sets the initial state of the simulated network transfer latency used by AVD. Latency is the delay in processing data across the network. The latency speed can also be adjusted in the emulator.

Screenshot from Android Studio Arctic Fox: Android Virtual Device (showing Network Latency options).

The network speed and latency options are based on data of interactions with networks in non-simulated environments at least because of their reliance on network standards developed to dictate the operation of real-world networks, such as LTE, HSDPA, UMTS, EDGE, GPRS, HSCSD, and GSM.

For instance, GSM corresponds to what is often referred to as the "2G" network standard established around 1991. Similarly, EDGE corresponds to Enhanced Data rates for GSM Evolution, an enhancement to GSM. And HSCSD corresponds to High Speed Circuit Switched Data, which is an enhancement to the data rate of circuit switched data in a GSM network. GPRS corresponds to a packet-oriented enhancement to 2G networks—General Packet Radio Service. UMTS is the Universal Mobile Telecommunications System, a new architecture that provided the basis for what is often referred to as the "3G" network standards. HSDPA, or High-Speed Downlink Packet Access, is an enhancement to the 3G network architecture to boost data capacity and improve download rates. Finally, LTE, or Long Term Evolution, represents the transition from 3G to what is typically referred to as "4G" network technology.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

1[C] wherein the network characteristics are based on data of interaction with networks in non-simulated environments.

Each of these network standards defines the general operation of the network, and these definitions provide theoretical constraints on the networks' capacity for communication, including bandwidth, latency, and speed constraints. These constraints can be further impaired based on network conditions, including the presence of physical obstacles, electro-magnetic interference, and/or distance between the base station and a mobile station with which it is communicating.

The development and evolution of these standards relied on data of interactions with real-world implementations of such networks at least for testing and proof-of-concept. Thus, Android Studio's speed and latency constraints correspond to each identified standard, which are based on data of interaction of networks in non-simulated environments.

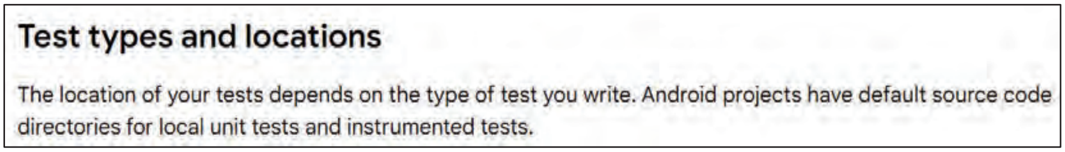
Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

8[A] The system of claim 1, wherein the software is further configured to create one or more scenarios that include scripts that impact either the performance of the application, or the network, or both.

Claim 8

8[A] The system of claim 1, wherein the software is further configured to create one or more scenarios that include scripts that impact either the performance of the application, or the network, or both.

Android Studio is configured to create one or more scenarios that include scripts that impact either the performance of the application, or the network, or both. Android Studio supports the creation of local unit tests and instrumented tests.

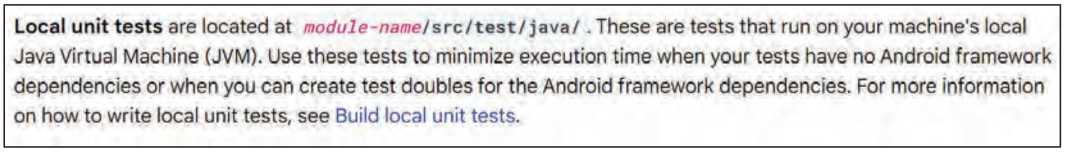


Test types and locations

The location of your tests depends on the type of test you write. Android projects have default source code directories for local unit tests and instrumented tests.

https://developer.android.com/studio/test/test-in-android-studio (last visited 5/1/2024).

Local unit tests test components of the application source code and therefore impact the performance of the application.



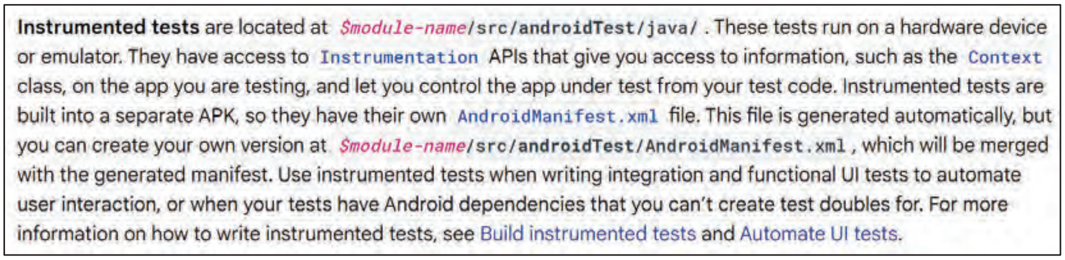
Local unit tests are located at `module-name/src/test/java/`. These are tests that run on your machine's local Java Virtual Machine (JVM). Use these tests to minimize execution time when your tests have no Android framework dependencies or when you can create test doubles for the Android framework dependencies. For more information on how to write local unit tests, see Build local unit tests.

https://developer.android.com/studio/test/test-in-android-studio (last visited 5/1/2024).

Instrumented tests run on the target device or an emulator of a target device. These tests "let you control the app under tests from your test code" and can be used to "automate user interaction."

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

8[A] The system of claim 1, wherein the software is further configured to create one or more scenarios that include scripts that impact either the performance of the application, or the network, or both.



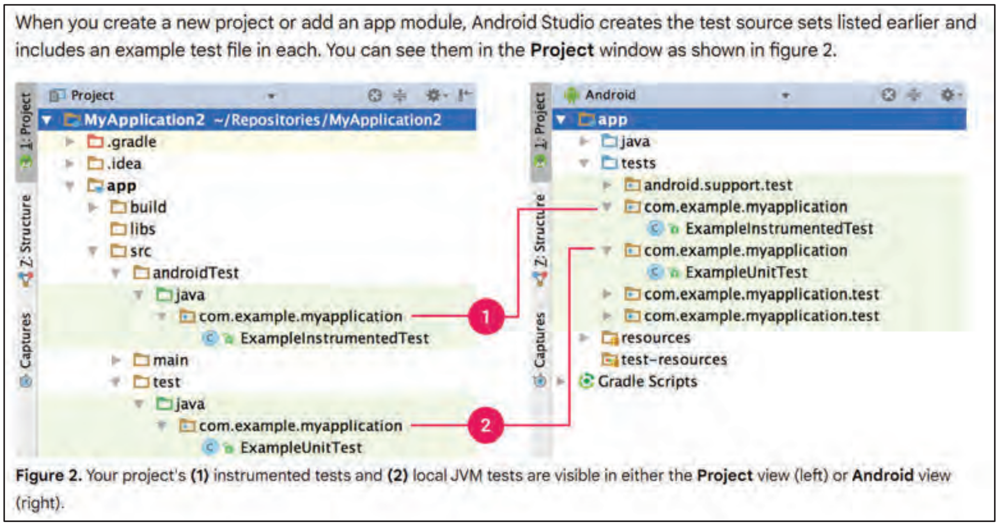
Instrumented tests are located at `$module-name/src/androidTest/java/`. These tests run on a hardware device or emulator. They have access to `Instrumentation` APIs that give you access to information, such as the `Context` class, on the app you are testing, and let you control the app under test from your test code. Instrumented tests are built into a separate APK, so they have their own `AndroidManifest.xml` file. This file is generated automatically, but you can create your own version at `$module-name/src/androidTest/AndroidManifest.xml`, which will be merged with the generated manifest. Use instrumented tests when writing integration and functional UI tests to automate user interaction, or when your tests have Android dependencies that you can't create test doubles for. For more information on how to write instrumented tests, see Build instrumented tests and Automate UI tests.

https://developer.android.com/studio/test/test-in-android-studio (last visited 5/1/2024).

Android Studio creates the project structure and sample tests to support both local unit tests and instrumented tests.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

8[A] The system of claim 1, wherein the software is further configured to create one or more scenarios that include scripts that impact either the performance of the application, or the network, or both.



When you create a new project or add an app module, Android Studio creates the test source sets listed earlier and includes an example test file in each. You can see them in the **Project** window as shown in figure 2.

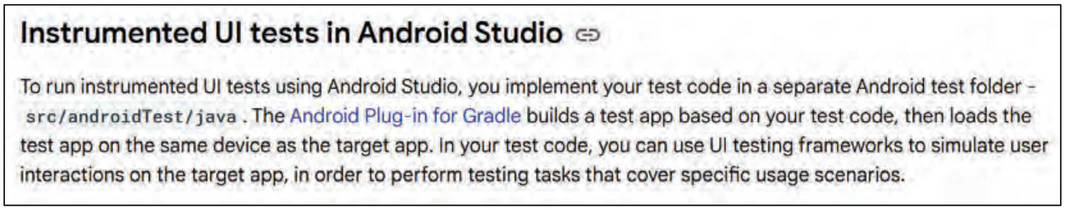
Figure 2. Your project's **(1)** instrumented tests and **(2)** local JVM tests are visible in either the **Project** view (left) or **Android** view (right).

https://developer.android.com/studio/test/test-in-android-studio (last visited 5/1/2024). These tests impact the performance of the application under tests and/or network.

Android Studio also supports UI tests which are scenarios that include scripts that impact either the performance of the application, or the network, or both.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

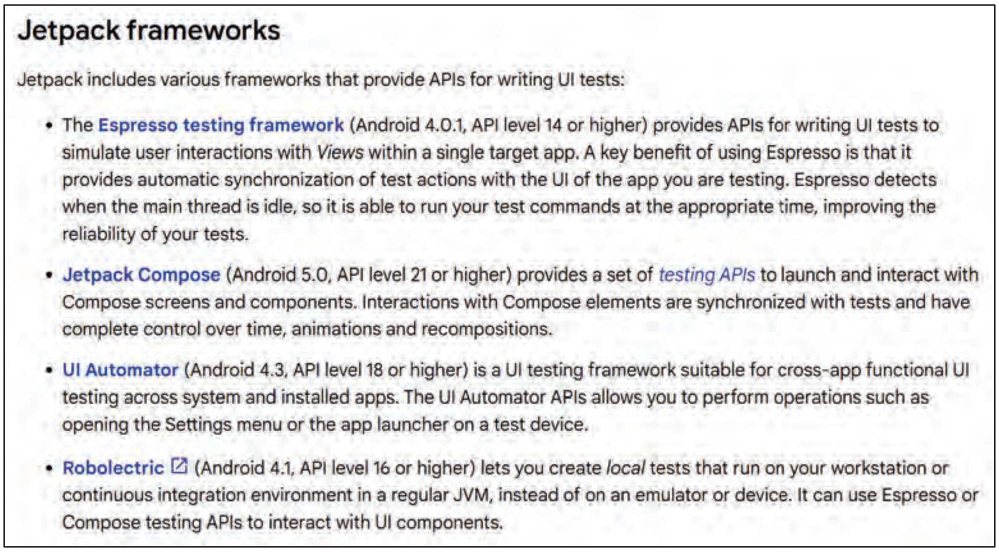
8[A] The system of claim 1, wherein the software is further configured to create one or more scenarios that include scripts that impact either the performance of the application, or the network, or both.



Instrumented UI tests in Android Studio

To run instrumented UI tests using Android Studio, you implement your test code in a separate Android test folder - `src/androidTest/java` . The Android Plug-in for Gradle builds a test app based on your test code, then loads the test app on the same device as the target app. In your test code, you can use UI testing frameworks to simulate user interactions on the target app, in order to perform testing tasks that cover specific usage scenarios.

https://developer.android.com/training/testing/instrumented-tests/ui-tests (last visited 5/2/2024). Android supports several APIs for writing UI test scenarios and scripts.



Jetpack frameworks

Jetpack includes various frameworks that provide APIs for writing UI tests:

- The **Espresso testing framework** (Android 4.0.1, API level 14 or higher) provides APIs for writing UI tests to simulate user interactions with *Views* within a single target app. A key benefit of using Espresso is that it provides automatic synchronization of test actions with the UI of the app you are testing. Espresso detects when the main thread is idle, so it is able to run your test commands at the appropriate time, improving the reliability of your tests.
- **Jetpack Compose** (Android 5.0, API level 21 or higher) provides a set of *testing APIs* to launch and interact with Compose screens and components. Interactions with Compose elements are synchronized with tests and have complete control over time, animations and recompositions.
- **UI Automator** (Android 4.3, API level 18 or higher) is a UI testing framework suitable for cross-app functional UI testing across system and installed apps. The UI Automator APIs allows you to perform operations such as opening the Settings menu or the app launcher on a test device.
- **Robolectric** ↗ (Android 4.1, API level 16 or higher) lets you create *local* tests that run on your workstation or continuous integration environment in a regular JVM, instead of on an emulator or device. It can use Espresso or Compose testing APIs to interact with UI components.

https://developer.android.com/training/testing/instrumented-tests/ui-tests (last visited 5/2/2024).

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

13[A] The system of claim 1, wherein the software is further configured to display data graphically which is configured to enable a user to identify either application performance, or network performance, or both.

Claim 13

13[A] The system of claim 1, wherein the software is further configured to display data graphically which is configured to enable a user to identify either application performance, or network performance, or both.

As discussed above for limitation 1[B].2 (and incorporated here by reference), the software is further configured to display data graphically which is configured to enable a user to identify either application performance, or network performance, or both. For example, the graphical displays in Android's Profilers (CPU, memory, network, energy) and Network Inspector enable a user to identify either application performance, or network performance, or both. *See* 1[B] (showing profile display windows that display data graphically for Android Profilers and Network Inspector).

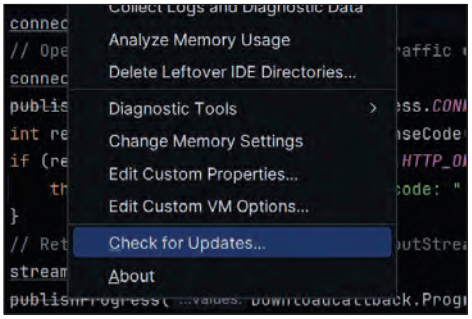
Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

14[A] The system of claim 1, wherein the software is further configured to interact with a network to enable a user to update the software

Claim 14

14[A] The system of claim 1, wherein the software is further configured to interact with a network to enable a user to update the software

Android Studio is configured to interact with a network to enable a user to update the software. Android Studio includes a "Check for Updates" option on its Help menu.

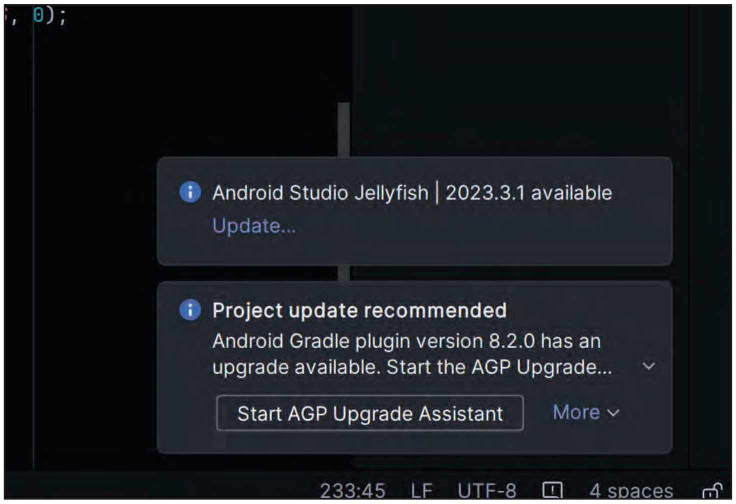

Collect Logs and Diagnostic Data
Analyze Memory Usage
Delete Leftover IDE Directories...
Diagnostic Tools
Change Memory Settings
Edit Custom Properties...
Edit Custom VM Options...
Check for Updates...
About

Screenshot from Android Studio Iguana: Help Menu Drop Down (showing "Check for Updates" option).

When updates are available for Android Studio, the user is notified of the update and given the option to download and install the update.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

14[A] The system of claim 1, wherein the software is further configured to interact with a network to enable a user to update the software



Android Studio Jellyfish | 2023.3.1 available
Update...

Project update recommended
Android Gradle plugin version 8.2.0 has an upgrade available. Start the AGP Upgrade...

Screenshot from Android Studio Iguana: Notifications (showing available Android Studio updates).

Claim 17

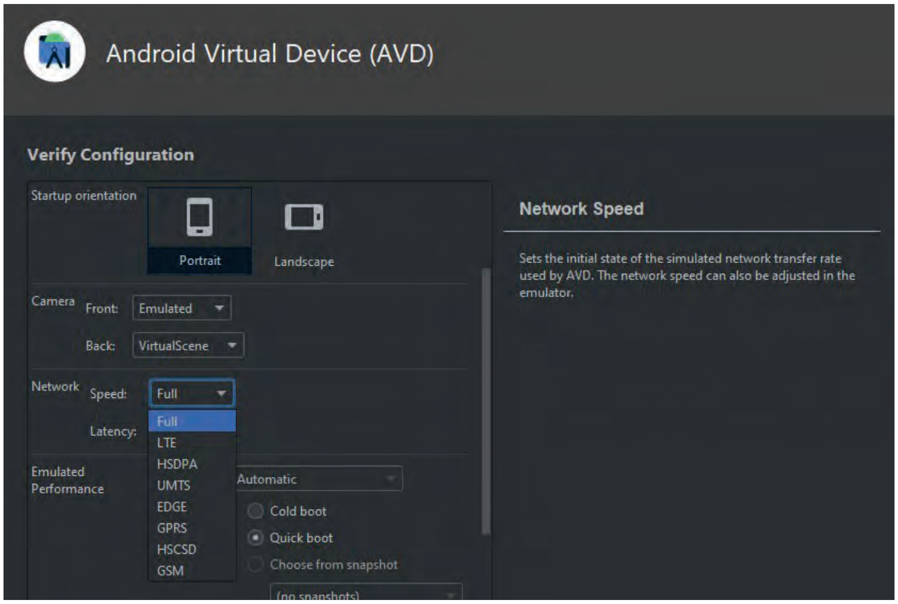
17[A] The system of claim 1, wherein the software is further configured to enable a user to select from one or more network characteristics for testing how well mobile content performs on the mobile device.

As discussed above for limitation 1[B].1 (and incorporated here by reference), the software is further configured to enable a user to select from one or more network characteristics (including, *inter alia*, network speed and network latency) for testing how well mobile content (such as that generated by the application being tested) performs on the mobile device.

The following figures show screenshots of the configuration screens for an Android Virtual Device. The first figure shows how an AVD can have a specific network speed associated with it, the speed options including: Full, LTE, HSDPA, UMTS, EDGE, GPRS, HSCSD, GSM.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

17[A] The system of claim 1, wherein the software is further configured to enable a user to select from one or more network characteristics for testing how well mobile content performs on the mobile device.



Android Virtual Device (AVD)

Verify Configuration

Startup orientation: Portrait

Camera Front: Emulated

Back: VirtualScene

Network Speed: Full

Latency:

Emulated Performance: Automatic

○ Cold boot

● Quick boot

○ Choose from snapshot

(no snapshots)

Network Speed

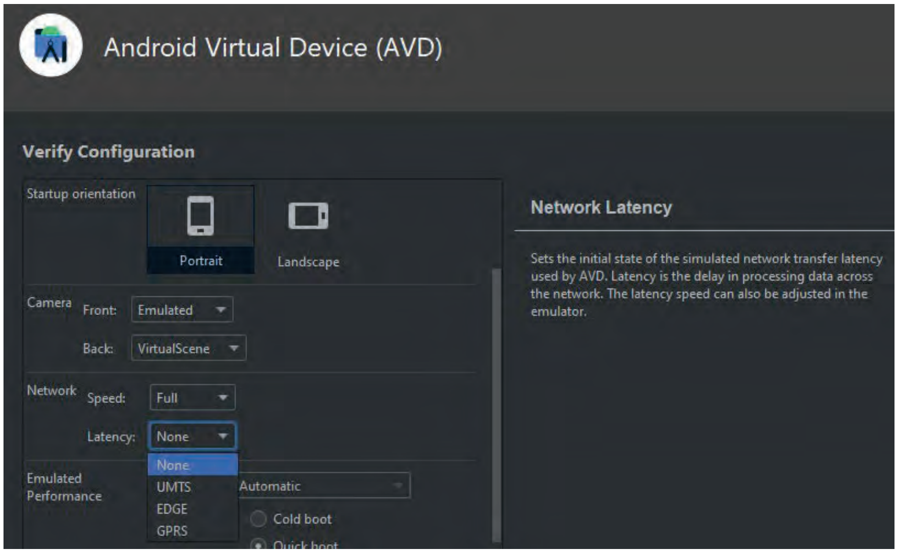
Sets the initial state of the simulated network transfer rate used by AVD. The network speed can also be adjusted in the emulator.

Screenshot from Android Studio Arctic Fox: Android Virtual Device (showing Network Speed options: Full, LTE, HSDPA, UMTS, EDGE, GPRS, HSCSD, GSM).

The following screenshot shows how an AVD can have a specific network latency associated with it, the latency options including, for example: None, UMTS, EDGE, GPRS.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

17[A] The system of claim 1, wherein the software is further configured to enable a user to select from one or more network characteristics for testing how well mobile content performs on the mobile device.



Android Virtual Device (AVD)

Verify Configuration

Startup orientation: Portrait

Camera Front: Emulated

Back: VirtualScene

Network Speed: Full

Latency: None

Emulated Performance: Automatic

○ Cold boot

● Quick boot

Network Latency

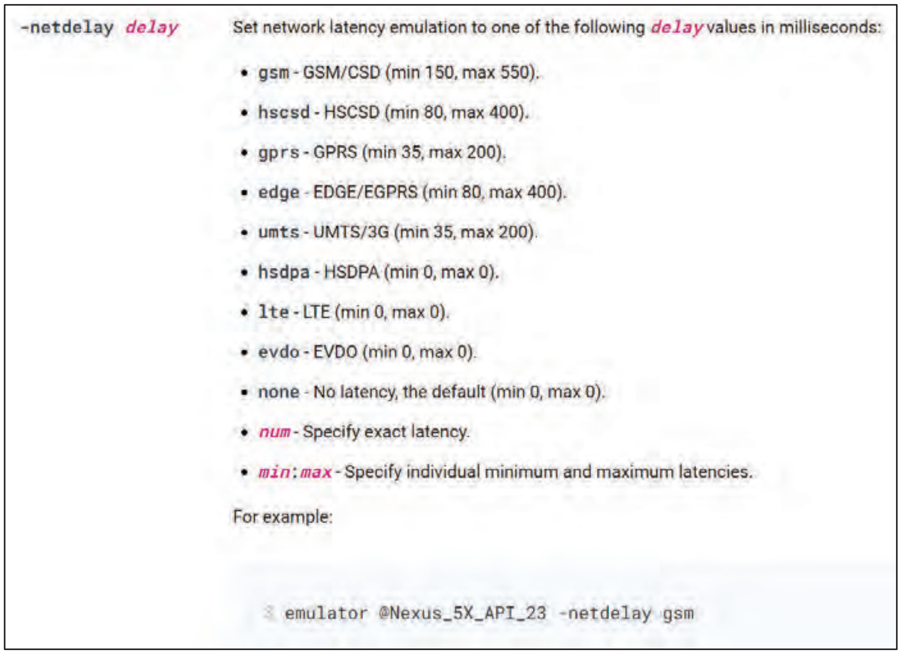
Sets the initial state of the simulated network transfer latency used by AVD. Latency is the delay in processing data across the network. The latency speed can also be adjusted in the emulator.

Screenshot from Android Studio Arctic Fox: Android Virtual Device (showing Network Latency options).

The network characteristics can also be controlled via the command-line options available for the Android Emulator, as illustrated below.

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

17[A] The system of claim 1, wherein the software is further configured to enable a user to select from one or more network characteristics for testing how well mobile content performs on the mobile device.



-netdelay *delay*

Set network latency emulation to one of the following *delay* values in milliseconds:

- gsm - GSM/CSD (min 150, max 550).
- hscsd - HSCSD (min 80, max 400).
- gprs - GPRS (min 35, max 200).
- edge - EDGE/EGPRS (min 80, max 400).
- umts - UMTS/3G (min 35, max 200).
- hsdpa - HSDPA (min 0, max 0).
- lte - LTE (min 0, max 0).
- evdo - EVDO (min 0, max 0).
- none - No latency, the default (min 0, max 0).
- *num* - Specify exact latency.
- *min*:*max* - Specify individual minimum and maximum latencies.

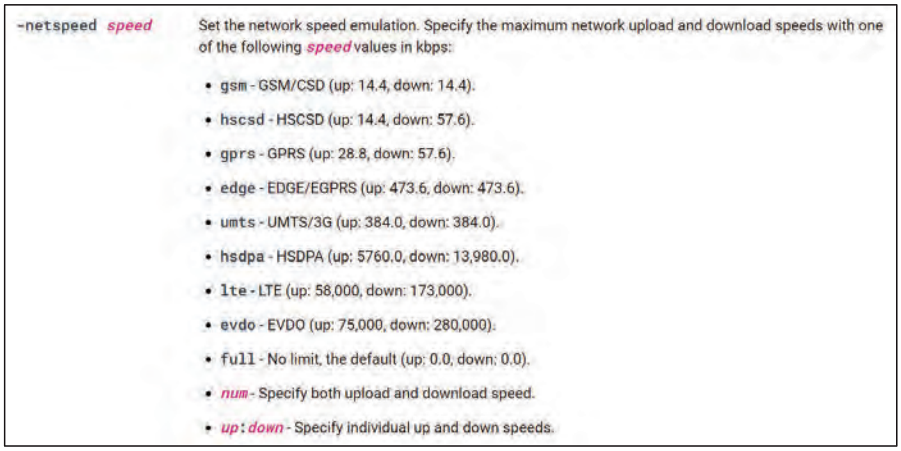
For example:

```
$ emulator @Nexus_5X_API_23 -netdelay gsm
```

https://developer.android.com/studio/run/emulator-commandline (last visited 4/7/2024) (showing options for setting network latency properties).

Infringement Chart for U.S. Patent No. 9,298,864 (Google's Android Studio Tools)

17[A] The system of claim 1, wherein the software is further configured to enable a user to select from one or more network characteristics for testing how well mobile content performs on the mobile device.



`-netspeed` *speed*

Set the network speed emulation. Specify the maximum network upload and download speeds with one of the following *speed* values in kbps:

- `gsm` - GSM/CSD (up: 14.4, down: 14.4).
- `hscsd` - HSCSD (up: 14.4, down: 57.6).
- `gprs` - GPRS (up: 28.8, down: 57.6).
- `edge` - EDGE/EGPRS (up: 473.6, down: 473.6).
- `umts` - UMTS/3G (up: 384.0, down: 384.0).
- `hsdpa` - HSDPA (up: 5760.0, down: 13,980.0).
- `lte` - LTE (up: 58,000, down: 173,000).
- `evdo` - EVDO (up: 75,000, down: 280,000).
- `full` - No limit, the default (up: 0.0, down: 0.0).
- *num* - Specify both upload and download speed.
- *up*:*down* - Specify individual up and down speeds.

https://developer.android.com/studio/run/emulator-commandline (last visited 4/7/2024) (showing options for setting network speed properties).